CPS3230 – Fundamentals of Software Testing

ASSIGNMENT PART 1 OF 3 – MARKET ALERT UM

UNIVERSITY OF MALTA
L-Università ta' Malta

Robert Fenech Adami
B.Sc. (Hons.) Information Technology (Software Development)
November 2022

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Robert Fenech Adami	[illegible]
Student Name	Signature
Student Name	Signature

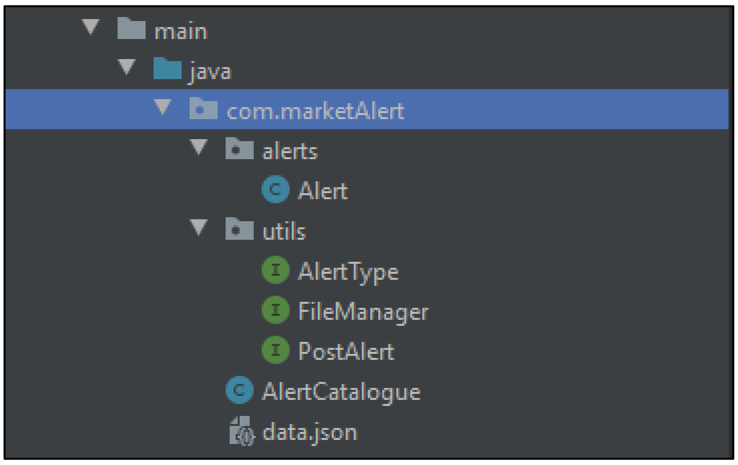
CPS3230	Assignment Part 1 of 3 – Market Alert UM
Course Code	Title of work submitted
13/11/2022	
Date	

Task 1

Github repository: https://github.com/robbyfa/marketAlertTask1

Google Drive:
https://drive.google.com/drive/folders/1wZVTt368XZxm4r4mASsn_p7OxxzBagyF?usp=sharing

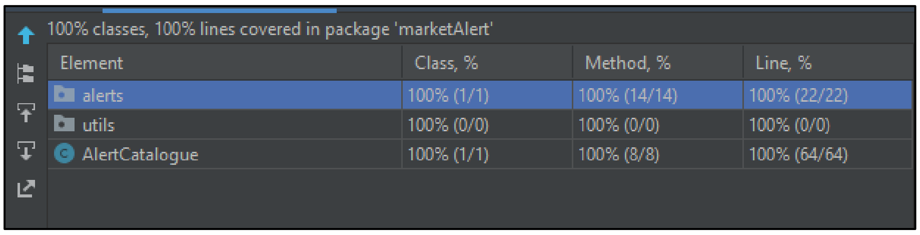
The SUT implemented consists of an 'Alert' object which includes all its relative details. An 'AlertCatalogue' class is implemented. This class contains a list of all alert objects created, together with relative methods to gather data through web scraping, publish alerts to marketAlertUM and, also, delete all alerts from the site, among other methods. A utilities folder which consists of several interfaces is also used. The architecture for the system under test can be seen in figure 1 below. All data was gathered from:
https://www.ultimate.com.mt/product-category/tv-audio/audio/headphones/



```
▼ main
  ▼ java
    ▼ com.marketAlert
      ▼ alerts
          Alert
      ▼ utils
          AlertType
          FileManager
          PostAlert
        AlertCatalogue
        data.json
```

[Fig.1]

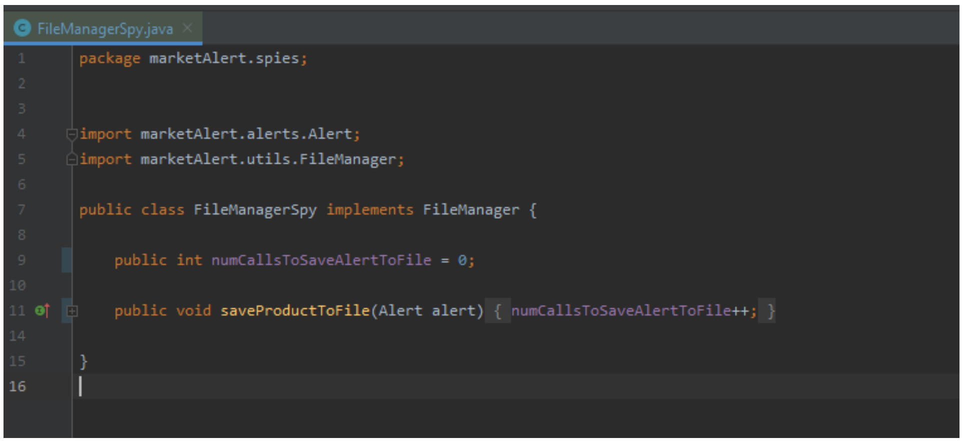
Several tests have been implemented for each aspect of the system using a test-driven approach, thus obtaining 100% test coverage, as seen in Figure 2 below. Each test was implemented using a 'setup, verify, exercise, and verify architecture; Furthermore, these tests also make use of spies in order to verify the number of alerts being saved to the alerts list and to verify the number of alerts being published to marketAlertUM via the REST API [fig. 3] [fig. 4]. The tests also make use of Mockito to assign the alert type to each alert being published.



100% classes, 100% lines covered in package 'marketAlert'

Element	Class, %	Method, %	Line, %
alerts	100% (1/1)	100% (14/14)	100% (22/22)
utils	100% (0/0)	100% (0/0)	100% (0/0)
AlertCatalogue	100% (1/1)	100% (8/8)	100% (64/64)

[Fig.2]

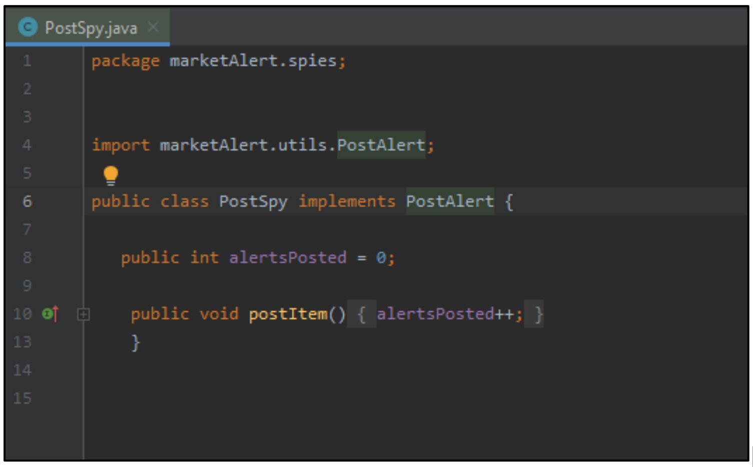

FileManagerSpy.java

```java
package marketAlert.spies;


import marketAlert.alerts.Alert;
import marketAlert.utils.FileManager;

public class FileManagerSpy implements FileManager {

    public int numCallsToSaveAlertToFile = 0;

    public void saveProductToFile(Alert alert) { numCallsToSaveAlertToFile++; }

}
```

[Fig.3]

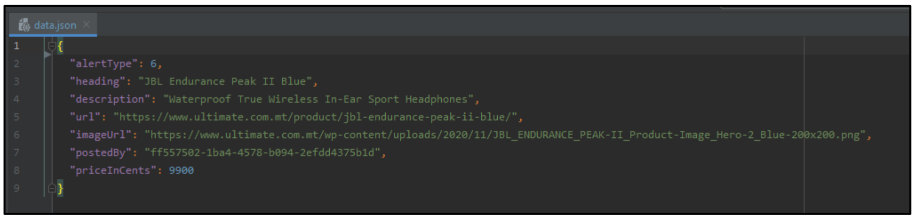
It is assumed that all data being gathered via the Chrome driver web scraper fall under the electronics category when publishing alerts via the REST API. Furthermore, it is assumed that all data, for 5 alerts, is being stored at one go, rather than adding and publishing one alert at a time. Further to this, the method used to call the web scraper simply implements a loop to gather alert data until the maximum limit is reached.

Another measure taken to ensure proper web scraping is the use of Gson and FileWriter dependency and library. These are used to store all data retrieved from the automated web scraper to a JSON file, as seen below in figure 5.


PostSpy.java

```java
package marketAlert.spies;


import marketAlert.utils.PostAlert;

public class PostSpy implements PostAlert {

    public int alertsPosted = 0;

     public void postItem() { alertsPosted++; }
     }


```

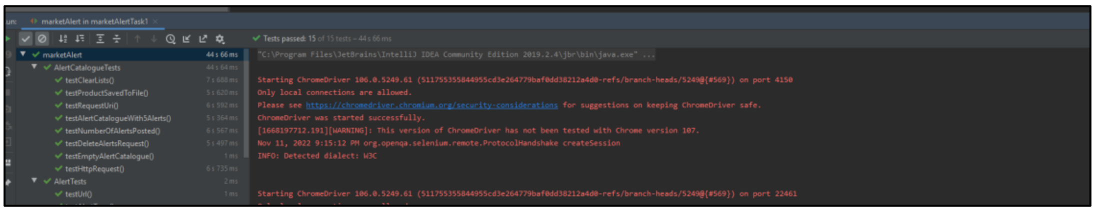
[Fig.4]


data.json

```json
{
  "alertType": 6,
  "heading": "JBL Endurance Peak II Blue",
  "description": "Waterproof True Wireless In-Ear Sport Headphones",
  "url": "https://www.ultimate.com.mt/product/jbl-endurance-peak-ii-blue/",
  "imageUrl": "https://www.ultimate.com.mt/wp-content/uploads/2020/11/JBL_ENDURANCE_PEAK-II_Product-Image_Hero-2_Blue-200x200.png",
  "postedBy": "ff557502-1ba4-4578-b094-2efdd4375b1d",
  "priceInCents": 9900
}
```

[Fig. 5]

The below figure shows the success status of all tests, with 15 of 15 passes.



```
Run:  marketAlert in marketAlertTask1

✓ Tests passed: 15 of 15 tests – 44 s 66 ms

▼ ✓ marketAlert                           44 s 66 ms
  ▼ ✓ AlertCatalogueTests                 44 s 64 ms
      ✓ testClearLists()                  7 s 688 ms
      ✓ testProductSavedToFile()          5 s 620 ms
      ✓ testRequestUri()                  6 s 592 ms
      ✓ testAlertCatalogueWith5Alerts()   5 s 364 ms
      ✓ testNumberOfAlertsPosted()        6 s 567 ms
      ✓ testDeleteAlertsRequest()         5 s 497 ms
      ✓ testEmptyAlertCatalogue()               1 ms
      ✓ testHttpRequest()                 6 s 735 ms
  ▼ ✓ AlertTests                                2 ms
      ✓ testUrl()                               1 ms
```

```
"C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.2.4\jbr\bin\java.exe" ...

Starting ChromeDriver 106.0.5249.61 (511755355844955cd3e264779baf0dd38212a4d0-refs/branch-heads/5249@{#569}) on port 4150
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1668197712.191][WARNING]: This version of ChromeDriver has not been tested with Chrome version 107.
Nov 11, 2022 9:15:12 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C


Starting ChromeDriver 106.0.5249.61 (511755355844955cd3e264779baf0dd38212a4d0-refs/branch-heads/5249@{#569}) on port 22461
```

AlertCatalogueTests:

Test Name	**Scope**
testEmptyAlertCatalogue	Assure alerts list is empty
testAlertCatalogueWith5Alerts	Assure web scraper gathers data for 5 alerts
testProductSavedToFile	Assure 5 alerts are saved to file
testNumberOfAlertsPosted	Assure number of alerts being posted
testHttpRequest	Assure successful use of API endpoint
testRequestUri	Assure proper HTTP request URI
testDeleteAlertsRequest	Assure successful use of API 'Delete' endpoint
testClearLists	Assure alerts list is emptied

AlertTests:

Test Name	**Scope**
testHeading	Assure heading variable is assigned
testDescription	Assure description variable is assigned
testUrl	Assure url variable is assigned
testImageUrl	Assure imageUrl variable is assigned
testPostedBy	Assure postedBy variable is assigned
testAlertType	Assure alertType variable is assigned
testPriceInCents	Assure priceInCents variable is assigned

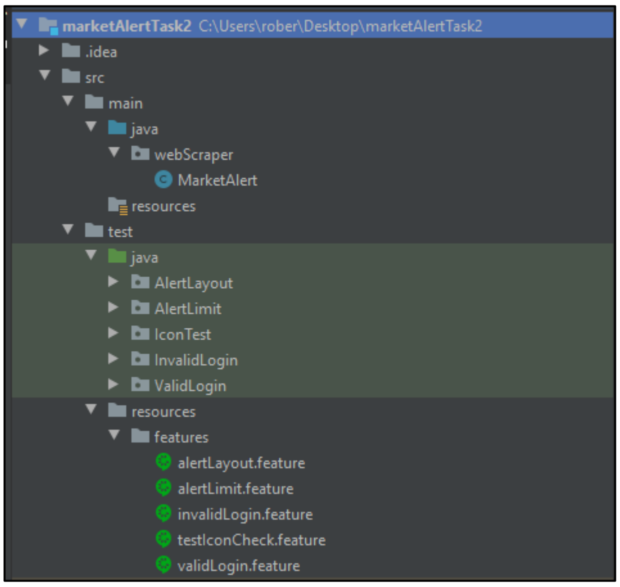
Name	**Scope**
FileManagerSpy	Used to store number of alerts saved
PostSpy	Used to store number of alerts being posted
AlertType (Mockito)	Used to set alert type for alerts

Task 2

Github repository: https://github.com/robbyfa/marketAlertTask2

Google Drive:
https://drive.google.com/drive/folders/1wZVTt368XZxm4r4mASsn_p7OxxzBagyF?usp=sharing

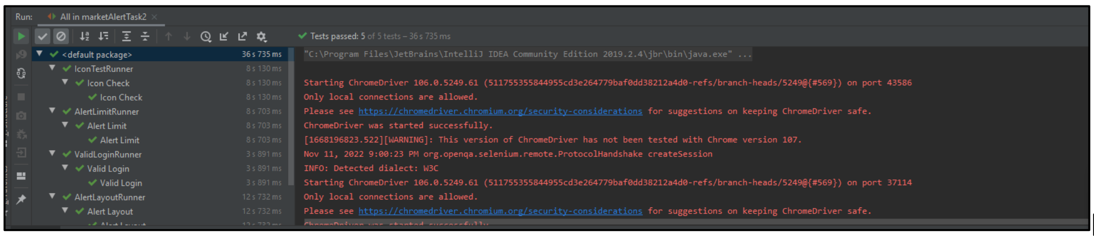
With regards to the SUT to be tested, all methods are declared within one 'MarketAlert' class. All feature files have been stored in the 'resources' folder in the test suite and all respective steps and runners have been stored in separate folders, i.e., the steps and runner for test1 are stored in the same package, while the steps and runner for test 2 are stored together in a different package. This was done to allow different features to have identical steps and thus avoid repetition of similar methods in the SUT. The project architecture can be seen below in Figure 6.



```
marketAlertTask2  C:\Users\rober\Desktop\marketAlertTask2
  .idea
  src
    main
      java
        webScraper
          MarketAlert
      resources
    test
      java
        AlertLayout
        AlertLimit
        IconTest
        InvalidLogin
        ValidLogin
      resources
        features
          alertLayout.feature
          alertLimit.feature
          invalidLogin.feature
          testIconCheck.feature
          validLogin.feature
```

[Fig. 6]

Upon running tests, as seen in figure 7 below, all 5 tests are successful.



```
Run:  All in marketAlertTask2
Tests passed: 5 of 5 tests – 36 s 735 ms

<default package>        36 s 735 ms
  IconTestRunner          8 s 130 ms
    Icon Check            8 s 130 ms
      Icon Check          8 s 130 ms
  AlertLimitRunner        8 s 703 ms
    Alert Limit           8 s 703 ms
      Alert Limit         8 s 703 ms
  ValidLoginRunner        3 s 891 ms
    Valid Login           3 s 891 ms
      Valid Login         3 s 891 ms
  AlertLayoutRunner      12 s 732 ms
    Alert Layout         12 s 732 ms
```

```
"C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.2.4\jbr\bin\java.exe" ...

Starting ChromeDriver 106.0.5249.61 (511755355844955cd3e264779baf0dd38212a4d0-refs/branch-heads/5249@{#569}) on port 43586
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1668196823.522][WARNING]: This version of ChromeDriver has not been tested with Chrome version 107.
Nov 11, 2022 9:00:23 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Starting ChromeDriver 106.0.5249.61 (511755355844955cd3e264779baf0dd38212a4d0-refs/branch-heads/5249@{#569}) on port 37114
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
```

[Fig. 7]

With regards to the testability of 'marketalertum.com', it is suggested that an alert counter element is implemented in the 'My Alerts' page to allow quicker verification when verifying the number of alerts published.

Valid Login Steps:

Label	**Step**	**Purpose**
Given	I am a user of marketalertum	Creates marketAlert instance
When	I login using valid credentials	Call valid login method
Then	I should see my alerts	Verify url

Invalid Login steps

Label	**Step**	**Purpose**
Given	I am a user of marketalertum	Creates marketAlert instance
When	I login using valid credentials	Call valid login method
Then	I should see the login screen again	Verify url

Alert Layout Steps:

Label	**Step**	**Purpose**
Given	I am a user of marketalertum	Creates marketAlert instance
When	I view a list of alerts	Call valid login method
Then / And	Each alert should contain a …	Find number of element

Alert Limit Steps:

Label	**Step**	**Purpose**
Given	I am an administrator of the website and I upload more than {int} alerts	Creates marketAlert instance and call addAlerts method.
Given	I am a user of marketalertum	Call marketAlert instance
When	I view a list of alerts	Call valid login method
Then	I should see {int} alerts	Count alerts and verify number